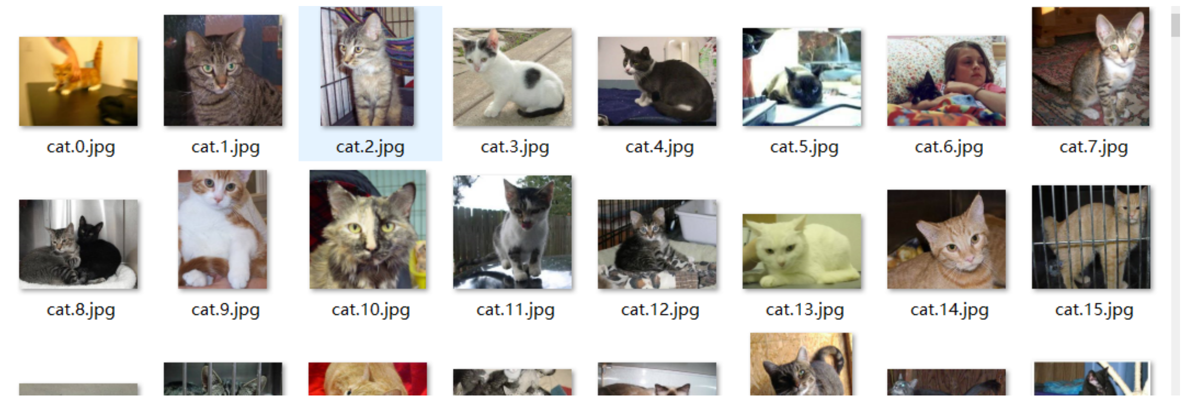
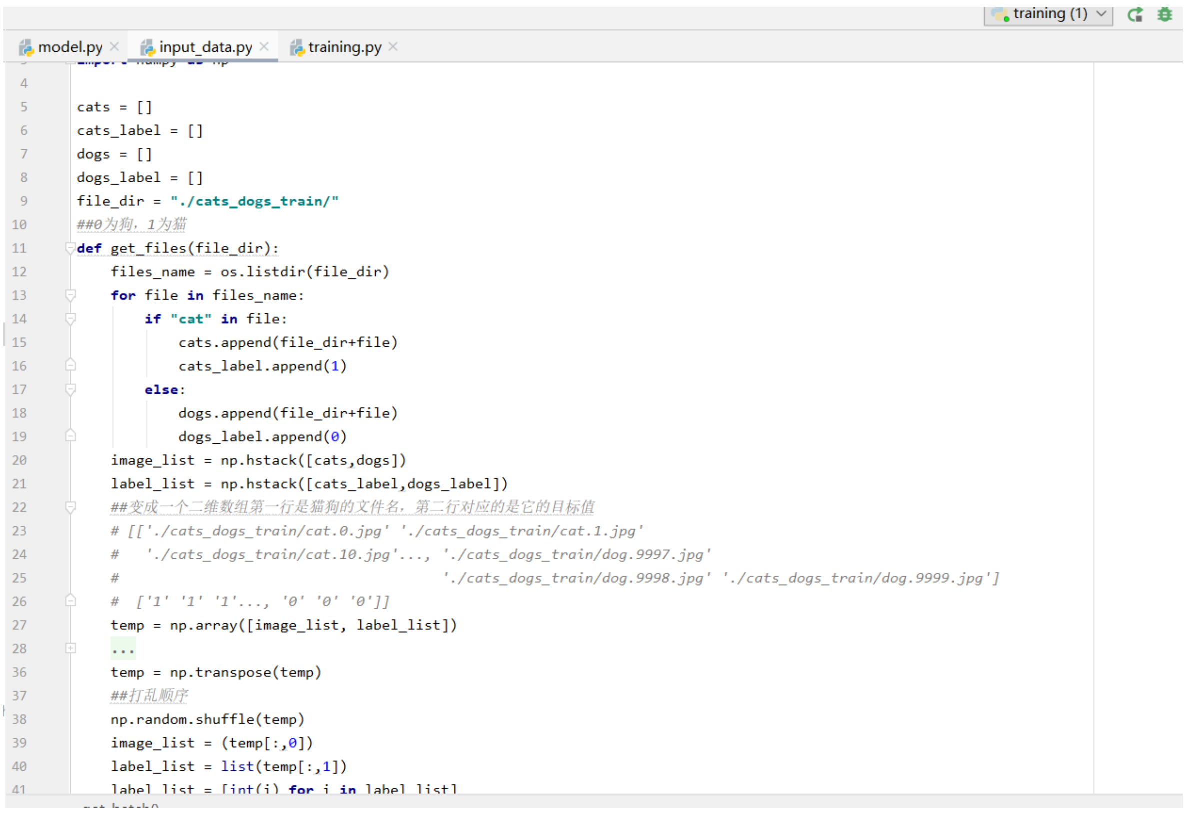
开发文档识别

1.首先在kaggle上下载下来训练数据



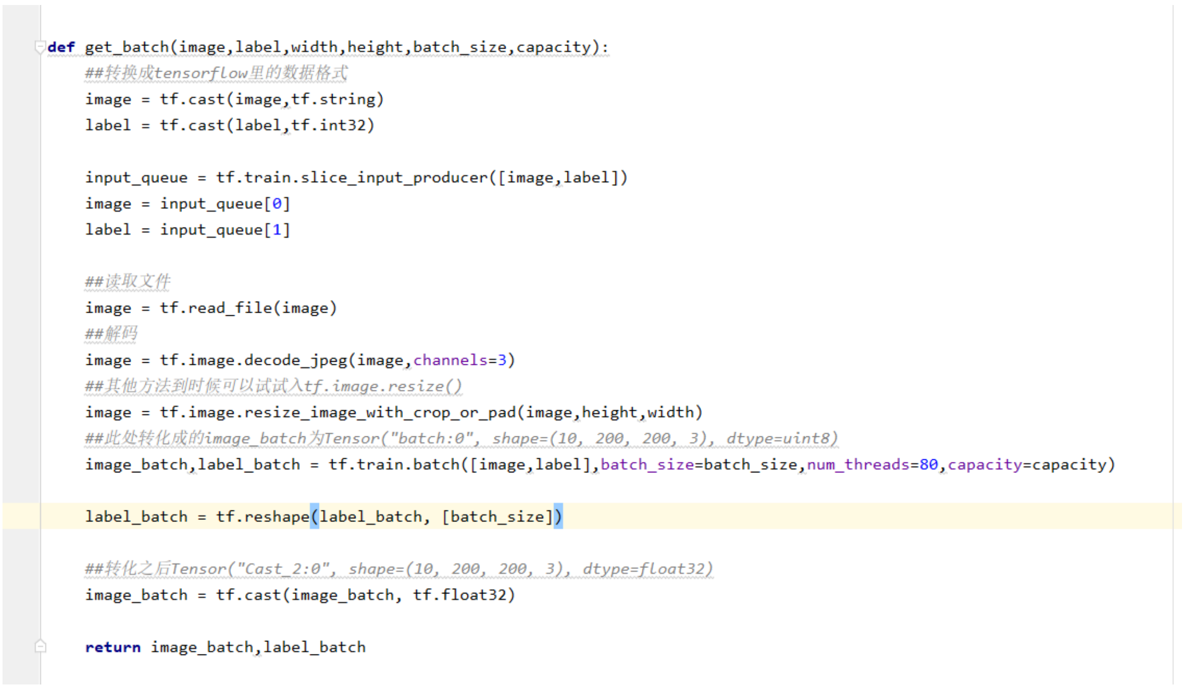
2.拿到之后对图片做一波处理。将样本与标签做一个一一对应



```python

cats = []
cats_label = []
dogs = []
dogs_label = []
file_dir = "./cats_dogs_train/"
##0为狗，1为猫
def get_files(file_dir):
    files_name = os.listdir(file_dir)
    for file in files_name:
        if "cat" in file:
            cats.append(file_dir+file)
            cats_label.append(1)
        else:
            dogs.append(file_dir+file)
            dogs_label.append(0)
    image_list = np.hstack([cats,dogs])
    label_list = np.hstack([cats_label,dogs_label])
    ##变成一个二维数组第一行是猫狗的文件名，第二行对应的是它的目标值
    # [['./cats_dogs_train/cat.0.jpg' './cats_dogs_train/cat.1.jpg'
    #   './cats_dogs_train/cat.10.jpg'..., './cats_dogs_train/dog.9997.jpg'
    #                                      './cats_dogs_train/dog.9998.jpg' './cats_dogs_train/dog.9999.jpg']
    #  ['1' '1' '1'..., '0' '0' '0']]
    temp = np.array([image_list, label_list])
    ...
    temp = np.transpose(temp)
    ##打乱顺序
    np.random.shuffle(temp)
    image_list = (temp[:,0])
    label_list = list(temp[:,1])
    label_list = [int(i) for i in label_list]
```

此函数返回值，就是样本及其对应的标签

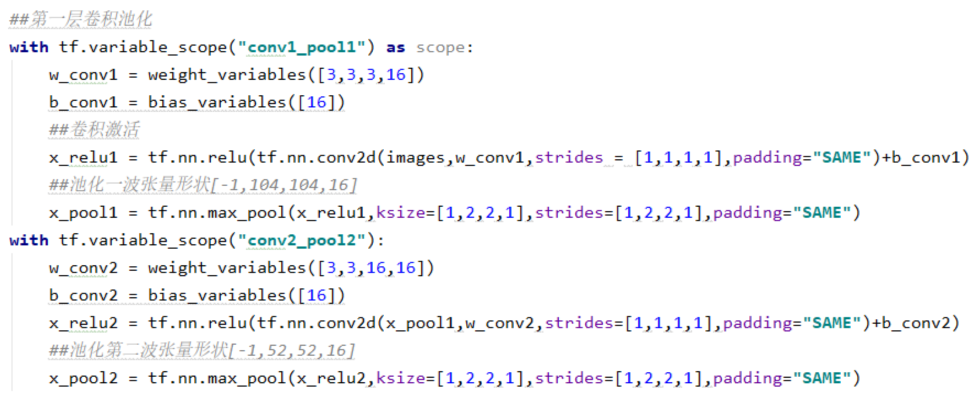
3.这一步是批量获取数据



```python
def get_batch(image,label,width,height,batch_size,capacity):
    ##转换成tensorflow里的数据格式
    image = tf.cast(image,tf.string)
    label = tf.cast(label,tf.int32)

    input_queue = tf.train.slice_input_producer([image,label])
    image = input_queue[0]
    label = input_queue[1]

    ##读取文件
    image = tf.read_file(image)
    ##解码
    image = tf.image.decode_jpeg(image,channels=3)
    ##其他方法到时候可以试试入tf.image.resize()
    image = tf.image.resize_image_with_crop_or_pad(image,height,width)
    ##此处转化成的image_batch为Tensor("batch:0", shape=(10, 200, 200, 3), dtype=uint8)
    image_batch,label_batch = tf.train.batch([image,label],batch_size=batch_size,num_threads=80,capacity=capacity)

    label_batch = tf.reshape(label_batch, [batch_size])

    ##转化之后Tensor("Cast_2:0", shape=(10, 200, 200, 3), dtype=float32)
    image_batch = tf.cast(image_batch, tf.float32)

    return image_batch,label_batch
```

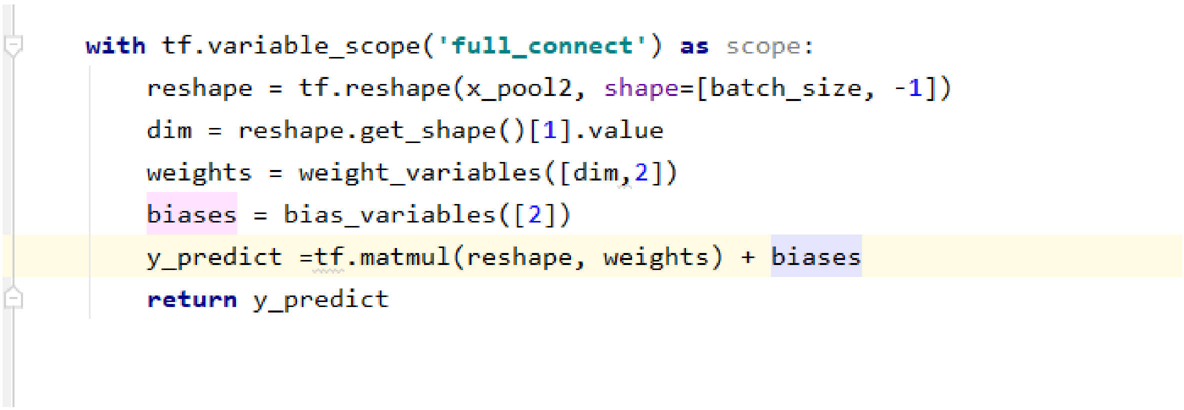
4.接下来定义模型

这里我定义了两层卷积池化



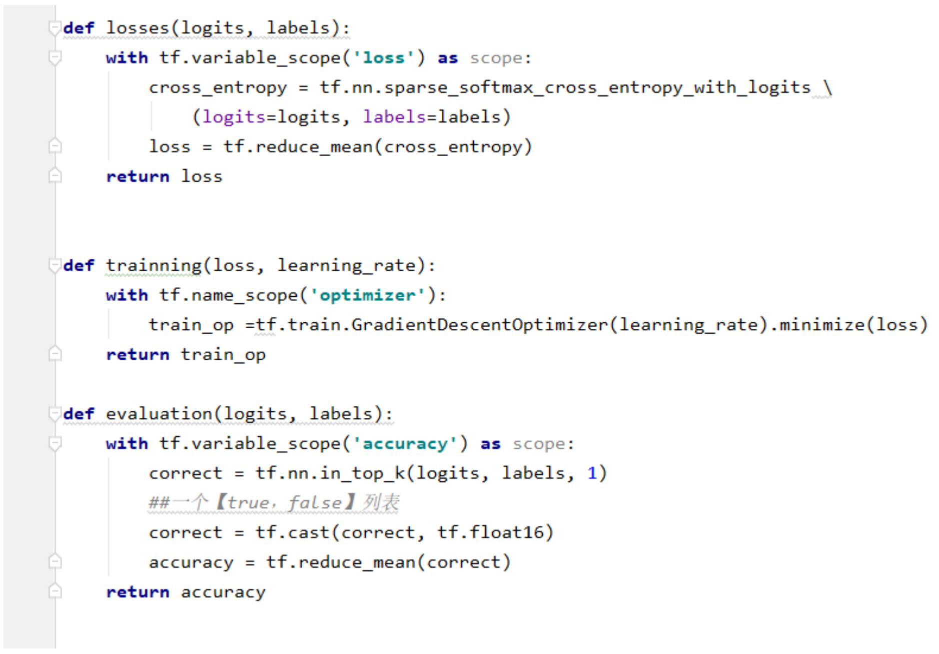
```python
##第一层卷积池化
with tf.variable_scope("conv1_pool1") as scope:
    w_conv1 = weight_variables([3,3,3,16])
    b_conv1 = bias_variables([16])
    ##卷积激活
    x_relu1 = tf.nn.relu(tf.nn.conv2d(images,w_conv1,strides = [1,1,1,1],padding="SAME")+b_conv1)
    ##池化一波张量形状[-1,104,104,16]
    x_pool1 = tf.nn.max_pool(x_relu1,ksize=[1,2,2,1],strides=[1,2,2,1],padding="SAME")
with tf.variable_scope("conv2_pool2"):
    w_conv2 = weight_variables([3,3,16,16])
    b_conv2 = bias_variables([16])
    x_relu2 = tf.nn.relu(tf.nn.conv2d(x_pool1,w_conv2,strides=[1,1,1,1],padding="SAME")+b_conv2)
    ##池化第二波张量形状[-1,52,52,16]
    x_pool2 = tf.nn.max_pool(x_relu2,ksize=[1,2,2,1],strides=[1,2,2,1],padding="SAME")
```

最后来一波全连接层



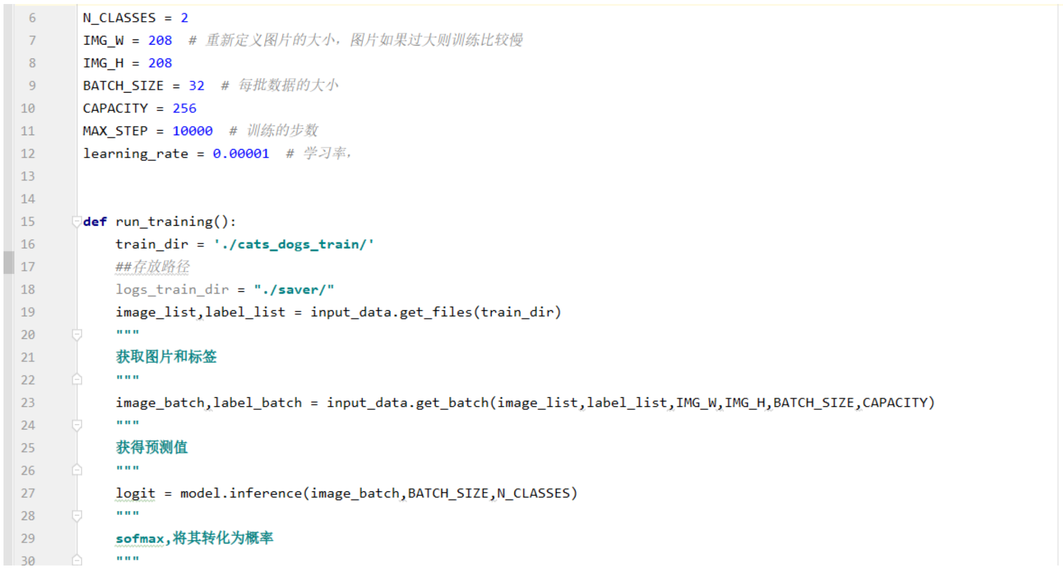
```python
with tf.variable_scope('full_connect') as scope:
    reshape = tf.reshape(x_pool2, shape=[batch_size, -1])
    dim = reshape.get_shape()[1].value
    weights = weight_variables([dim,2])
    biases = bias_variables([2])
    y_predict =tf.matmul(reshape, weights) + biases
    return y_predict
```

评估和训练

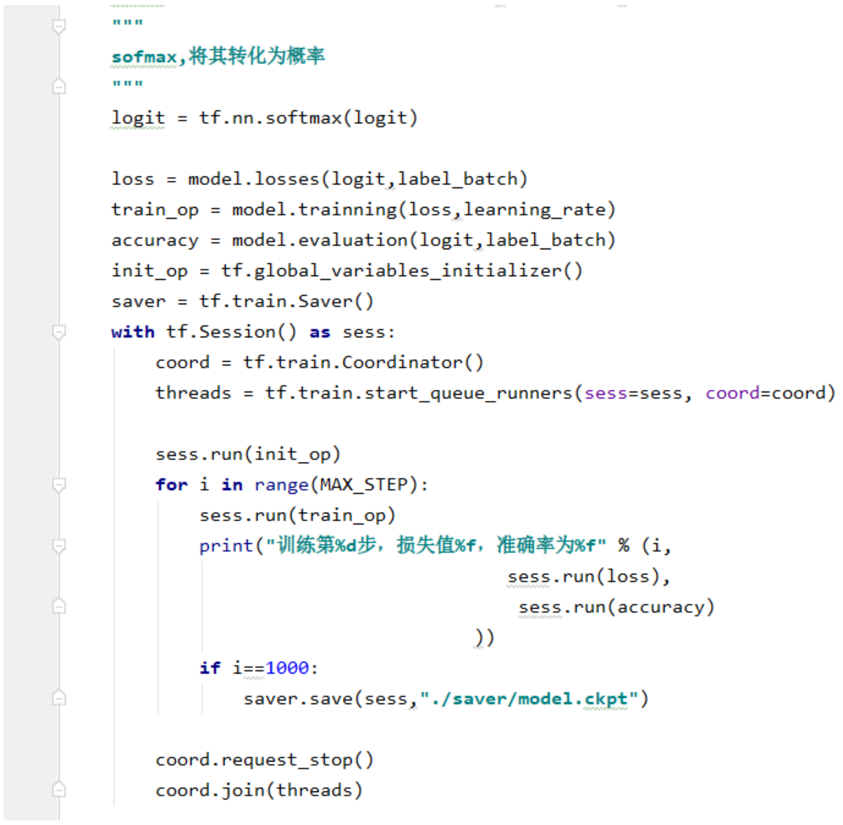


```python
def losses(logits, labels):
    with tf.variable_scope('loss') as scope:
        cross_entropy = tf.nn.sparse_softmax_cross_entropy_with_logits \
            (logits=logits, labels=labels)
        loss = tf.reduce_mean(cross_entropy)
    return loss


def trainning(loss, learning_rate):
    with tf.name_scope('optimizer'):
        train_op =tf.train.GradientDescentOptimizer(learning_rate).minimize(loss)
    return train_op

def evaluation(logits, labels):
    with tf.variable_scope('accuracy') as scope:
        correct = tf.nn.in_top_k(logits, labels, 1)
        ##一个【true，false】列表
        correct = tf.cast(correct, tf.float16)
        accuracy = tf.reduce_mean(correct)
    return accuracy
```

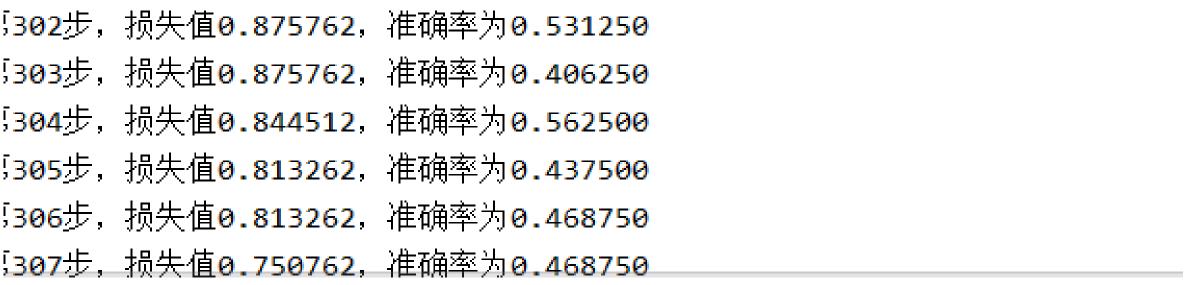
接下来就可以开始训练了



```python
N_CLASSES = 2
IMG_W = 208  # 重新定义图片的大小，图片如果过大则训练比较慢
IMG_H = 208
BATCH_SIZE = 32  # 每批数据的大小
CAPACITY = 256
MAX_STEP = 10000  # 训练的步数
learning_rate = 0.00001  # 学习率，


def run_training():
    train_dir = './cats_dogs_train/'
    ##存放路径
    logs_train_dir = "./saver/"
    image_list,label_list = input_data.get_files(train_dir)
    """
    获取图片和标签
    """
    image_batch,label_batch = input_data.get_batch(image_list,label_list,IMG_W,IMG_H,BATCH_SIZE,CAPACITY)
    """
    获得预测值
    """
    logit = model.inference(image_batch,BATCH_SIZE,N_CLASSES)
    """
    sofmax,将其转化为概率
    """
```



```
"""
sofmax,将其转化为概率
"""
logit = tf.nn.softmax(logit)

loss = model.losses(logit,label_batch)
train_op = model.trainning(loss,learning_rate)
accuracy = model.evaluation(logit,label_batch)
init_op = tf.global_variables_initializer()
saver = tf.train.Saver()
with tf.Session() as sess:
    coord = tf.train.Coordinator()
    threads = tf.train.start_queue_runners(sess=sess, coord=coord)

    sess.run(init_op)
    for i in range(MAX_STEP):
        sess.run(train_op)
        print("训练第%d步，损失值%f，准确率为%f" % (i,
                                              sess.run(loss),
                                               sess.run(accuracy)
                                            ))
        if i==1000:
            saver.save(sess,"./saver/model.ckpt")

    coord.request_stop()
    coord.join(threads)
```



```
302步，损失值0.875762，准确率为0.531250
303步，损失值0.875762，准确率为0.406250
304步，损失值0.844512，准确率为0.562500
305步，损失值0.813262，准确率为0.437500
306步，损失值0.813262，准确率为0.468750
307步，损失值0.750762，准确率为0.468750
```

准确率很低